WHAT IS CLAIMED IS:

1. A distributed processing system comprising:
(a) a plurality of discrete, autonomous program processes, wherein each program process is capable of generating destinationless messages;
(b) at least one data repository for receiving and storing destinationless messages generated by program processes;
wherein each program process is associated with configuration information, which includes information for associating a program process with a data repository in at least one of a read and write capacity.

2. The distributed processing system according to claim 1, wherein each discrete autonomous program process is a module.

3. The distributed processing system according to claim 1, wherein each data repository is a channel.

4. The distributed processing system according to claim 1, wherein the configuration information for each program process is utilized at a runtime step to establish one of a read and write relationship with a data repository.

5. A reconfigurable data processing system comprising:
(a) a plurality of discrete, autonomous program processes, wherein each program process is capable of the generation of destinationless messages;
(b) at least one data repository for receiving and storing destinationless messages generated by program processes;
(c) a plurality of configuration parameters associating program processes with at least one data repository in one of a read and write capacity, the plurality of configuration parameters establishing a data flow mechanism.

6. A data processing system comprising:
(a) a plurality of modules, wherein a module is a discrete process and is associated with a capability for generating destinationless messages;

(b) at least one data repository, wherein each data repository receives and stores destinationless messages generated by modules;

(c) an API ("Application Program Interface"), wherein the API provides at least one function for associating a read operation of a module with a data repository and a write operation of a module with a data repository.

7. The data processing system according to claim 6, wherein each module does not include programming instructions relating to an interaction with a second module.

8. The data processing system according to claim 6, wherein each module is associated with configuration information, wherein the configuration information includes information for associating the module with a data repository for writing destinationless messages.

9. The data processing system according to claim 6, wherein each module is associated with configuration information, wherein the configuration information includes information for associating the module with a data repository for reading destinationless messages.

10. The data processing system according to claim 6, further including a second API for defining a module, wherein the module API includes at least one of a function for initializing, working and terminating a module.

11. The data processing system according to claim 6, further including a third API for defining a structure for a message, wherein the structure includes at least one field.

12. The data processing system according to claim 6, wherein a destinationless message is a BLOB ("Binary Large Object").

13. The data processing system according to claim 8, wherein a module is associated with a data repository for writing messages at runtime utilizing the configuration information.

14. The data processing system according to claim 9, wherein a module is associated with a data repository for reading destinationless messages at runtime utilizing the configuration information.

15. The data processing system according to claim 6, wherein each of the data repositories is a segment of memory.

16. The data processing system according to claim 6, wherein each module includes a plurality of programming instructions including programming instructions for at least one of remotely initializing the module and remotely terminating the module.

17. The data processing system according to claim 16, wherein the programming instructions for remotely initializing the module include a function pointer.

18. A system for application development, comprising:

(a) a module API wherein the module API includes a plurality of function calls for defining a module, wherein a module is a discrete process that includes a capability for a generation of destinationless messages;

(b) a channel API, wherein the channel API includes a plurality of function calls for at least one of opening, writing to and reading from a channel, wherein a channel stores destinationless messages generated by modules;

(c) a message API, wherein the message API includes a plurality of function calls for defining a structure of destinationless messages.

19. The system according to claim 18, wherein the module API, channel API and message API are provided as statically linked libraries.

20. A method for developing a computer program comprising the steps of:

(a) defining a plurality of program modules, wherein each module includes a process to perform an associated function, and wherein each module includes a capability for generating destinationless messages;

(b) defining a plurality of communication pathways between the modules in order to achieve a desired program behavior;

(c) at runtime, resolving the communication pathways to associate the plurality of program modules with a plurality of data repositories, wherein the data repositories receive and store messages.

21. The method according to claim 20, wherein each of the plurality of program modules is associated with a data repository in one of a read and write capacity.

22. The method according to claim 20, wherein step (c) further includes the step of specifying configuration information for each module with respect to the plurality of data repositories in one of a read and write capacity as a function of the communication pathways.

23. A method for application development comprising the steps of:

(a) defining at least one module, wherein a module includes a discrete code element and performs a functional behavior;

(b) defining an interaction configuration between the at least one module to achieve an application behavior, wherein the interaction configuration includes re-configurable communication pathways for an exchange of messages between a first module and a second module;

(c) defining a deployment configuration for the application, wherein the deployment configuration defines at least a relationship between each module and one or more hosts within a computing environment;

(d) deploying the modules to the computing environment as a function of the deployment configuration; and,

(e) establishing the communication pathways between the at least one module as a function of the interaction and the deployment configuration.

24. The method according to claim 23, wherein step (e) further includes the steps of:

(i) defining at least one decentralized channel; and,

(ii) establishing communication between a first module and a second module sharing a communication pathway by setting the first module to exchange messages with the second module on a decentralized channel.

25. The method according to claim 24, wherein a decentralized channel is a shared memory resource including at least one of a resource on a storage device and a random access memory ("RAM").

26. The method according to claim 23, wherein computing environment is a computer network.

27. The method according to claim 23, wherein step (e) further includes the steps of:

(i) defining at least one decentralized channel;

(ii) if a first module and a second module sharing a communication pathway are designated to reside on separate hosts as part of the deployment configuration, deploying a sender module to a first host associated with the first module and deploying a receiver module to a second host associated with the second module, wherein the sender module and the receiver modules perform communications between the first and second host; and,

(iii) establishing a communication between the first module and the sender module and establishing a communication between the second module and the receiver module.

28. A program module comprising:

(a) an initialization function process, wherein the initialization function process is executed upon an initialization of the program module;

(b) a work function process, wherein the work function process exhibits a behavior associated with the program module; and,

(c) a termination function process, wherein the termination function process is executed upon a termination of the program module.

29. The program module according to claim 28, wherein the initialization function process, the work function process and the termination function process are callback functions.

30. A distributed application development system comprising a processor, wherein the processor is adapted to:
provide a graphical user interface ("GUI") wherein the GUI:
(a) provides for a specification of a plurality of discrete, autonomous program processes to be used in the application, and receives information relating thereto;

(b) provides for a specification of a plurality of communication pathways between the plurality of program processes and receives information relating thereto; and,

(c) provides for a specification of a deployment configuration and receives information relating thereto;

wherein the processor is further adapted to modify runtime configuration information relating each of the program processes as a function of the information received by the GUI.

31. The distributed application development system according to claim 30, wherein the runtime configuration information of each program process includes information relating to associating the program process with a data repository in at least one of a read and write capacity.

32. The distributed application development system according to claim 31, wherein the processor updates the configuration information of each program process associating the program process with a data repository as a function of the communication pathways specified in step (b).